Faster, More Practical, and Higher Quality Unstructured Light Field Photography

ANDREW ZHAO, Carnegie Mellon University, USA

Unstructured light field photography is a practical compromise between structured light field photography, which is equipment-intensive, and traditional digital photography, which does not capture as much scene information as structured light field photography. An implementation for refocusing images, as described in class as assignment 4, involves (1) capturing a planar video, (2) computing x-y shifts using template matching to align one object to the same position in all the frames, and (3) merging the stack of frames with a simple sum. This approach suffers from practicality and ease-of-use issues due to the need to select a template with particular conditions, and may produce results that poorly approximate defocus blur due to artifacts from the capture process. I propose and implement a Structure-from-Motion-based alignment procedure to improve on practicality, and a set of 3 merging algorithms based on weighting images in the stack to lessen artifacts. SfM-based alignment meets the intended goals and also introduces major performance improvements, one of the merging algorithms produces improved results, and the other two leave open areas for study.

ACM Reference Format:
Andrew Zhao. 2023. Faster, More Practical, and Higher Quality Unstructured Light Field Photography. 1, 1 (December 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BACKGROUND

1.1 Unstructured Light Field Photography

Light Field photography is a very interesting and useful photography technique that enables richer results (refocusing, 3D reconstruction, etc.) than traditional photography by capturing extra dimensions of light from a given scene. However, sampling the 4D domain of a light field typically requires very specialized equipment to take 2D photographs at fixed intervals or known positions [Levoy and Hanrahan 1996]. Another approach, called unstructured light field photography, allows light field capture from consumer cameras by using scene information to recover camera poses and correlate data between photographs [Davis et al. 2012].

A pipeline for amateur light field photography, with the aim of refocusing capabilities, was introduced in class as the last part of assignment 4. The pipeline involves the following steps:

(1) Capture a series of planar image samples of a scene, with 2D movement in both axes in the plane
(2) Pick an object of focus in the scene, and compute an x-y shift for each image that to place the object in the same image coordinates across all frames
(3) Merge the stack of shifted images

This process, performed on a series of all-in-focus or relatively-focused samples, aims to produce a resulting image akin to a photograph focused on the object with an aperture proportional to the amount of movement in step (1). Importantly, refocusing is trivial for unstructured light fields since the displacements are known, but this unstructured capture method requires special techniques to compute image-specific shifts. Assignment 4 also introduces one particular implementation:

- Compute image shifts by template-matching on the focus point
- Merge stack by performing a normalized sum

Such a method is relatively simple to implement, and produces sufficient results where the selected object is in focus and surrounding areas are defocused. However, this method suffers from several limitations that both limit the usefulness of this method in practice and cause the quality of produced results to fall behind refocused images from structured light fields and true 2D images captured by a lens-based camera.

1.2 Limitations and Issues

Some limitations both discussed and seen in the results of the aforementioned implementation are as follows:

(I1) Refocusing requires manual selection of a template location and template size.
(I2) Following on the above, this method can only refocus on locations with unique, non-repeating appearances.
(I3) The shape of the planar camera motion is visible as artifacts in the refocused image.
(I4) Under planar capture assumptions, varying camera velocity causes certain areas of blurring to be more or less intense than others.
(I5) Planar assumptions are difficult to uphold in practice; even under planar assumptions, rotation in the z-axis/roll-axis is possible and negatively affects quality.

These limitations are with respect to the benchmark of digital photographs captured via a camera with lens, which we aim to replicate in the refocusing procedure. Lens defocus blur is generally approximated as a Gaussian blur [Liu et al. 2021], which I establish as a baseline and use as inspiration towards methods of improving on the raised issues.

2 PIPELINE IMPROVEMENTS

In this project, I aim to improve the unstructured light field photography process by addressing practicality and quality issues. Of important note is that I plan to revise the *implementation only*, and not the pipeline. This is because the abstract pipeline is a good definition of a procedure to generate refocused images while accounting for low equipment requirements. Furthermore, the biggest issues

Author's address: Andrew Zhao, Carnegie Mellon University, PA, 15213, USA, andrewhzhao@gmail.com.

ACM XXXX-XXXX/2023/12-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

seen in section 1.2 are implementation-specific, and not related to the pipeline.

2.1 Alignment Step

Firstly, many downsides of the current implementation arise from the choice of template-based alignment (Issues I1, I2). Among adjacent works in the field that require image alignment or 3D reconstruction in unstructured capture conditions, SLAM [Levoy and Hanrahan 1996] and Structure from Motion (SfM) [Agarwal et al. 2011] techniques are popular. Notably, Structure from Motion is a well-known technique that applies feature matching to create correspondences of 2D points across several images of the same object, and applies stereo geometry principles to deduce 3D scene geometry, along with camera poses and intrinsic parameters in the process. As such, SfM is in some ways an extension of the current implementation that applies feature matching to the in-focus template, except that it improves accuracy by considering all features in the scene and by refining estimates through known geometric constraints. SfM usually relies on a fixed set of images and iteratively refines its 3D estimations, while SLAM is popular for real-time applications as the set of image data grows over time. Accordingly, I plan to apply SfM to improve step (2) of the pipeline, the alignment computation. The predicted advantages are as follows:

- Streamline the refocusing process, by eliminating the need for manual template selection
- Improve robustness, by expanding the range of patterns and features that can be accurately focused on
- Improve algorithmic efficiency of refocusing several times with one image stack

Furthermore, SfM opens the door to many other possibilities to improve quality that are out of scope for this project:

- Allow non-planar capture, by using the computed camera intrinsics and extrinsics to reproject all samples to be planar
- Compensate for rotation error in the capture process using computed camera poses

2.2 Merging Step

The unstructured nature of the capture process causes several quality issues in refocused results (I3, I4). Figure 1 shows an example of the issues described in I3, I4 in a refocusing with a very large effective aperture, using the naive merging method of simply summing and normalizing the shifted image stack. Keeping the abstract pipeline in mind, such artifacts can be compensated for in the merging step. The inputs to this step are a series of frames and their x/y coordinate shifts to align the stack about an in-focus object, and the output is a refocused image. Accordingly, I propose several weighted merging schemes that assign each image a weight based on some properties about its shift, in hopes of reducing the aforementioned artifacts and getting closer to a gaussian blur in defocused regions.

Following the idea of a Gaussian blur, the most obvious improvement would be to weight samples according to their straight-line shift distance according to a normal distribution's PDF. In effect, this would be convolving an image of the shifts-per-image graph by a Gaussian kernel, which should hopefully lessen the degree to which the sampling path appears as an artifact in the results.

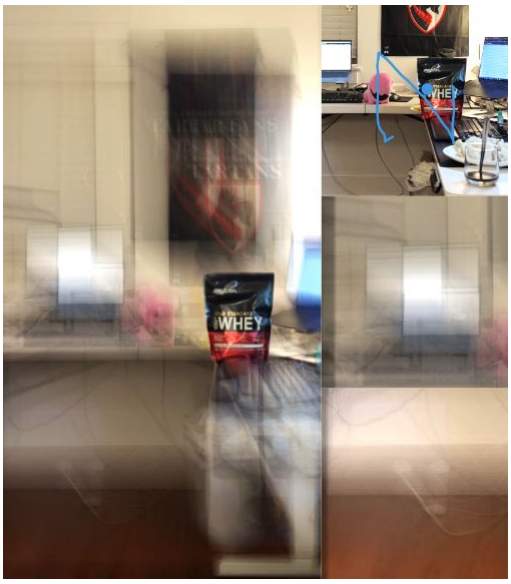

Fig. 1. Example template-based refocusing, and associated artifacts. Left: refocused image. Top: Motion path that the camera took to capture these frames, shaped like the letter N. Middle: blurred laptop screen, showing four distinctive corners (I4). Bottom: Power outlet, blurred in an N-shape (I3).

Another issue that is not accounted for is varying sample density. Weighting samples based on the local density, or inversely to the average distance to neighbors, would ideally help solve this. However, it is not immediately obvious how density or distance should be mathematically defined. One idea is to construct a spline from the motion path, since frames are sequential, and compensate for distance along the spline; however, this suffers from density variation in the spline's shape itself. A better idea is based on the fact that defocus blur is radially symmetric. If the shifts are viewed in polar coordinates, the definition of "neighbor" becomes trivial, and we can weight points based on the θ-distance to their left and right neighbors. The hope is that this would compensate for directional density; for example, if 3 samples lie in one sector while 1 lies in a different one, the 3 samples should be each weighted 1/3 with respect to the 1.

Overall, the above ideas show that this problem reduces to approximating a uniform sampling from a set of discrete samples of arbitrary shape. Concretely, these ideas are distilled into 4 methods for implementation:

(M1) Uniform weighting: Benchmark/control, uniformly weight all images in the stack
(M2) Gaussian weighting: Weight images based on straight-line shift distance using a Uniform PDF, reducing the impact of outliers and blurring the sampling shape
(M3) Radial weighting: Weight images inversely to their local densities, expressed as angular density.
(M4) Hybrid weighting: With each shift in polar form, apply Gaussian weighting with the radius and Radial weighting with the direction. Hopefully combines the best of both.

Under this framework, the original naive method is equivalent to (M1) uniform weighting.

3 METHODS

3.1 Pipeline Implementation

The overall pipeline for the project was implemented in Python, based on my assignment 4 base code, but highly modified to support fast and efficient file I/O and modularity. Frames are extracted from video inputs, which are run through an alignment implementation of choice (Template/SfM), which takes in an image stack and specified focus point and outputs per-image integer shifts. The merging function accepts an image stack, the corresponding shifts, and a weighting scheme, and produces the final refocused image.

3 Structure from Motion implementations were considered for this project:

- Alicevision Meshroom: An extensively built-out and feature-rich photogrammetry library. Meshroom is written in and supposedly can be used easily as a Python library, but the overall package, which is built primarily for dense 3D reconstruction, ended up being overkill for camera pose estimation.
- OpenMVG: A minimal SfM library consisting of several compact binaries. Ended up being difficult to build and setup.
- ColMap: An SfM/MVG library similar to OpenMVG. The executable was much easier to setup and use, but unfortunately does not integrate with Python.

Ultimately, ColMap proved the most practical to use. I used the Python pipeline I wrote to extract frames from videos, passed the videos into ColMap, which estimated 3D feature points and camera poses, and loaded those results back into Python for the merging step.

After obtaining ColMap results, an additional step is required to compute 2D shift data from 3D pose data. The 3D pose data is shared by all possible refocus choices, but due to perspective projection, the 2D image shifts will vary depending on the point of focus. I considered 2 ideas for this step:

- Select an arbitrary point in 3D space, and use the camera intrinsics and extrinsics computed by ColMap to project this point into 2D coordinates in each image's space. The shifts can be computed by translating these coordinates such that the center of the cluster is (0, 0). A possible refocusing interface would be to display a frame, query a user mouse click and desired distance, and compute the 3D point by moving the specified distance down a ray projected through the clicked 2D point.
- Allow the user to select an SfM-computed feature, for which the depth and thus 3D coordinate is known.

The first idea has a much heavier implementation, and requires an extra degree of input from the user. However, it has the advantage of being able to focus on any point in space, even those without textures or features. Since one of the goals of this project was to eliminate the template-picking step, the second method has an advantage in ease-of-use. Overall, I ended up realizing that ColMap actually outputs its points as a graph, mapping 3D points to the corresponding 2D point in every image, and likewise in the reverse direction. As a result, alignment computing reduces from computing projections to a simple graph traversal. Thus, with the potential benefit of extremely fast refocusing using existing data, I chose to implement the second idea. In the final implementation, the user can simply select any point, and the refocused image is generated within seconds.

Fig. 2. Sample result screenshot from ColMap. Note that the distinctive N-shaped sample pattern is very accurately extracted, and the red/black Carnegie Mellon banner visible in the backdrop of other figures can be seen reconstructed sparsely.

Specific parameters include a sigma value of $\sigma = 0.35$ for Gaussian weighting, a maximum angle difference cutoff of $\frac{\pi}{8}$ for Radial weighting, and a 75% image match requirement for features to be focus points. On this last point, each 3D point/feature from SfM is correlated with 2D points in corresponding images, are not always guaranteed to match every image. Accordingly, selected refocus points are filtered for those that are found in at least 75% of the stack, to have a sufficient number of frames in the refocus. I found that this threshold consistently resulted in a very feature-dense output.

3.2 Data Collection

For evaluation purposes, I took several video segments of the same scene, moving the camera in the N-shape recommended in assignment 4 and illustrated in 1, top right. Evaluating across multiple capture shapes would be ideal, but given the time constraints, I focused on analyzing 3 different sizes of N capture patterns. Overall, the analysis focused on running the 3 data sets through permutations of the 2 alignment methods and 4 proposed weighting schemes. Runtime data was collected for programs running a Dell XPS15 9500, with an Intel i7-10750H processor, Nvidia 1650Ti GPU, and 16GB RAM.

4 RESULTS + DISCUSSION

4.1 SfM Alignment

Switching Alignment to SfM resulted in several substantial improvements. First, the practicality/ease-of-use was greatly improved:

- Template-matching: User must specify a template x/y position and its x/y dimensions, and wait a substantial amount of time for the alignment-computation step before seeing the result.
- SfM: Poses are pre-computed, and afterwards, user selects a focus point. Shifts are computed very quickly from pose data, and the result is displayed, with minimal input.

In addition, the robustness has been improved, as the SfM alignment is somewhat capable of focusing near textureless or otherwise challenging areas (3). In the example in Figure 3, the algorithm finds the closest threshold-meeting feature point as the point of focus, resulting in an approximate result. However, an implementation using the second projection idea described in section 3.1 would truly allow this capability as the 3D point at the roof could be specified.

Finally, I observed a substantial runtime improvement that was not foreseen. Figure 5 shows sample runtimes to generate image shifts for Template Matching vs SfM; SfM is much faster overall. In addition, Template Matching is very quick (<30 seconds in most cases) to pre-process, but requires several minutes for each refocus; this is not good for the most common use case, which is to see different refocusings of one image stack. On the other hand, SfM takes a few minutes to compute camera poses, but a single-digit number of seconds to refocus, which is orders of magnitudes better here. With multiple refocusings, the advantage of SfM becomes stronger. This is likely because refocusing is just a single graph traversal step per image in $O(1)$, and it is far faster than a template correlation process.

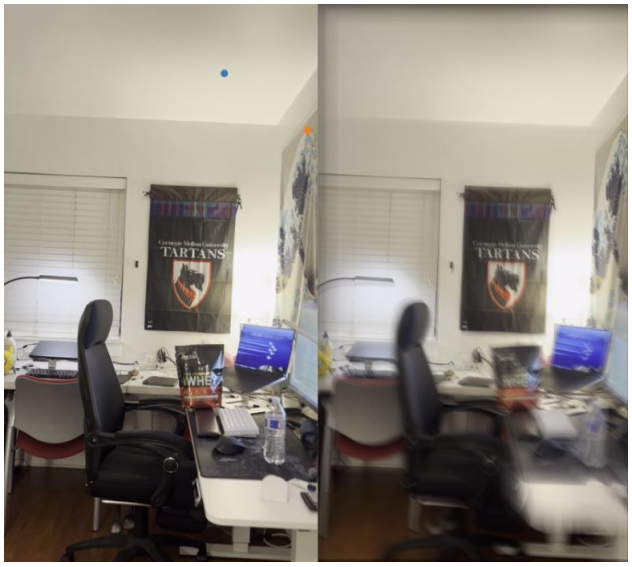

Fig. 3. Example result from focusing on the featureless ceiling. The select focus point is blue, and the selected SfM feature is orange.

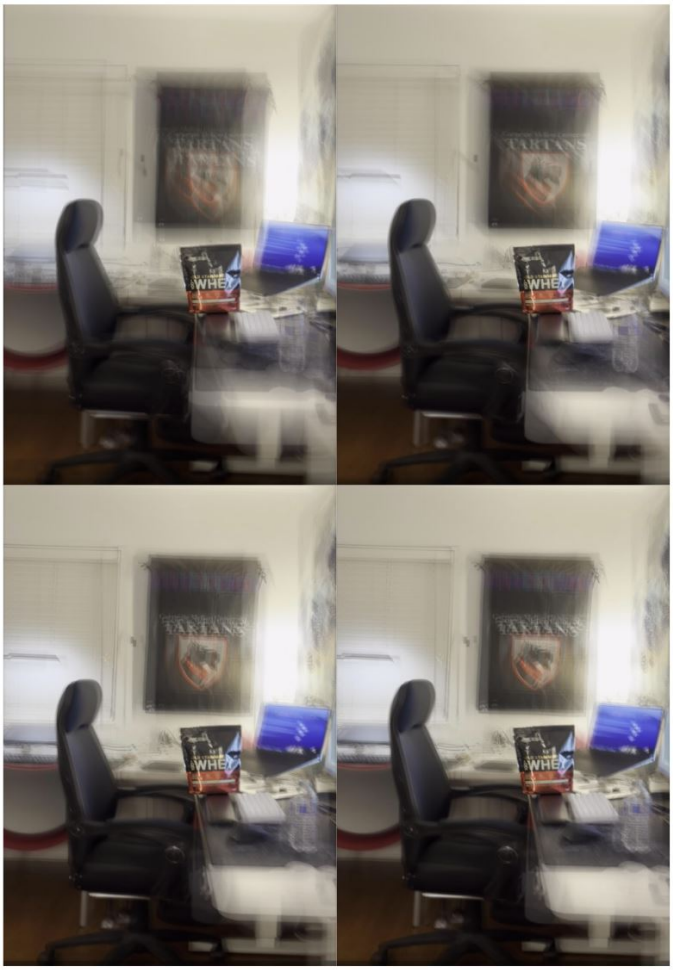

Fig. 4. Refocused results using the 4 weighting schemes. Clockwise from top left: Uniform, Gaussian, Hybrid, Radial.

Other factors that may account for this difference include that ColMap is very optimized, as I was running the provided CUDA version that specifically takes advantage of NVidia GPUs. However, this doesn't affect SfM's extremely fast Python runtime, nor does it change the aforementioned principle that SfM is very efficient to compute further refocusings.

4.2 Weighted Stack Merging

The Gaussian weighting scheme produced excellent results, while the radial and hybrid schemes did not perform well on the examined data set with an N-shaped movement pattern.

- **Gaussian**: Gaussian weighting had the anticipated positive effects of smoothing outliers and "blurring" the sampling path shape. The sigma value had a strong influence on the result; as expected, high sigma values tend towards the Dirac delta function, which in our case reduces the stack to the single image with the lowest shift distance. Low sigma values degenerate to the Uniform weighting case. 0.35 sigma

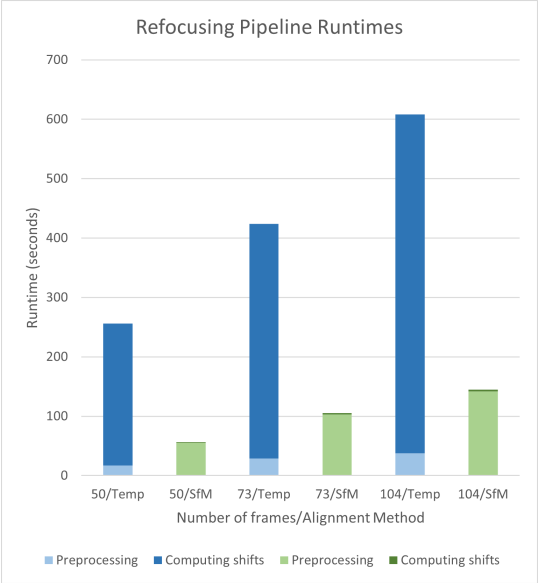

Fig. 5. Runtime comparison between Template matching (blue) and SfM (green) methods, distinguishing between preprocessing (1-time per stack) and refocusing (1-time per refocus) operations. Proportionally, template matching spends less time preprocessing but more time computing shifts. SfM is much faster overall.

performed well in every aperture setting, but it remains to be seen whether this performs well across widely varying focal lengths and other scenes. Plotting weight values versus frame index (Figure 6) further confirms the desired behavior; the sharp corners around the N, at indices 15/40, are under-weighted, while the straight edges are weighted more heavily, especially the components closer to the centroid of the point cluster.

- **Radial**: The visual effect achieved from radial weighting was replacing a few dense artifacts with many, lesser artifacts. The reason behind this is visible in Figure 6, which shows large spikes in a few areas and somewhat flat but noisy patterns in other areas. The original implementation actually had far higher spikes, but capping the θ-difference to $\frac{\pi}{8}$ mitigated this effect. It seems that both outliers and noise are dominating in this weighting scheme, which overall does not produce good results.
- **Hybrid**: This scheme unfortunately combines the worst of both Radial and Gaussian schemes, rather than the best of both. Noise and outliers are very prominent, and the defocusing effect is reduced in all areas, causing the result to look closer to its inputs and not a refocused image.

5 CONCLUSIONS

In this project, I developed improvements to the alignment computation and stack merging steps of the unstructured lightfield refocusing process introduced in class. I showed that using SfM

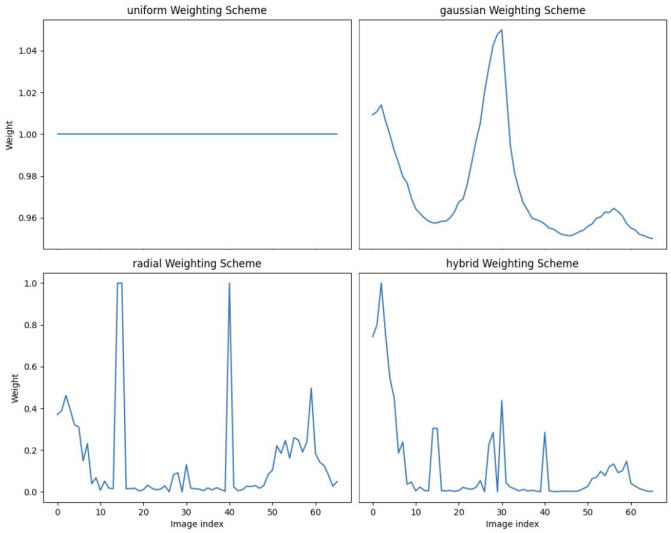

Fig. 6. Image weight vs. image index in video sequence, for the 4 different weighting schemes. Progressing from left to right is equivalent to tracing the shape of an N-curve; the corners appear to be around frame indices 15 and 40.

instead of template-based alignment greatly decreases runtime, improves ease of use, introduces more robustness, and leaves the door open for further unexplored improvements. I also demonstrated that merging the shifted stack using image weights based on a normal distribution on the shift radius delivers substantial benefits to refocused image quality, while the theorized polar-based weighting methods performed poorly.

6 FURTHER WORK

These findings have opened several areas of potential further study, on different topics:

- **Result evaluation.** Plotting weights with respect to position along the movement spline was a good step to unveil additional patterns that couldn't be deduced from the refocus results. However, defocus quality can be further evaluated through sharpness analysis of results.
- **Validate accuracy improvements.** In section 2 I theorize that SfM has higher feature matching accuracy since it corroborates all features in an image; it may be worth testing and validating this point.
- **More weighting schemes.** Radial and hybrid weighting did not perform well, but I believe the polar-based theory below them to be sound. Further investigation may be able to arrive at new and better schemes based on a polar view of the pattern.
- **Full Python integration.** While the pipeline is much more streamlined, it is still a step away from requiring only a file input and focus point specification since files must be copied back and forth from ColMap. Using a Python-compatible SfM library may be able to achieve this.

REFERENCES

Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. 2011. Building rome in a day. *Commun. ACM* 54, 10 (2011), 105–112.

Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured light fields. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 305–314.

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 31–42. https://doi.org/10.1145/237170.237199

Yu-Qi Liu, Xin Du, Hui-Liang Shen, and Shu-Jie Chen. 2021. Estimating Generalized Gaussian Blur Kernels for Out-of-Focus Image Deblurring. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 3 (2021), 829–843. https://doi.org/10.1109/TCSVT.2020.2990623